IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS
MARSHALL DIVISION

UNILOC 2017, LLC,

Plaintiff,

v.

GOOGLE LLC,

Defendant.

NO. 2:18-cv-00497-JRG-RSP
NO. 2:18-cv-00501-JRG-RSP
NO. 2:18-cv-00551-JRG-RSP

Filed Under Seal

DECLARATION OF RYAN LOVELESS IN SUPPORT OF UNILOC'S OPPOSITION TO GOOGLES' MOTION FOR SUMMARY JUDGMENT OF NON-INFRINGEMENT AND IN SUPPORT OF UNILOC'S FRCP 56(d) REQUEST FOR DISCOVERY

I, Ryan Loveless, do declare and state as follows:

1. I am counsel of record for Plaintiff Uniloc 2017, LLC ("Uniloc") in the above-captioned matters. I submit this declaration in support of Uniloc's Opposition to Google's Motion for Summary Judgment of Non-Infringement and in support of Uniloc's request under Fed. R. Civ. P. 56(d) for discovery before the Court rules on that motion.

2. Attached hereto as **Attachment 1** is a true and correct copy of relevant excerpts of the open264.org home page stating "Cisco has taken their H.264 implementation and open sourced it…" found at http://www.openh264.org (last accessed on November 19, 2019).

3. Attached hereto as **Attachment 2** is a true and correct copy of relevant excerpts of the Chrome Status Platform page stating "H.264 software encoder/decoder in Chrome for WebRTC … Include a H.264 video codec encoder and decoder in Chrome for use with WebRTC… The plan is to use OpenH264…" found at https://www.chromestatus.com/feature/6417796455989248 (last accessed on November 19, 2019).

4. Attached hereto as **Attachment 3** is a true and correct copy of relevant excerpts of Google public source code for WebRTC stating "Enable this to build OpenH264 encoder" found at https://chromium.googlesource.com/external/webrtc/+/fb11424551dae924869ae54059cb1612836cb6f7/webrtc/build/webrtc.gni (last accessed November 19, 2019) and OpenH264 source code found at https://chromium.googlesource.com/external/github.com/cisco/openh264/+/4c74cc8fdeba12f8556528aca6f12352331cab67/codec/decoder/core/inc/fmo.h (last accessed November 19, 2019) showing use of file "fmo.h", "Flexible Macroblock Ordering implementation" and "\brief Initialize Wels Flexible Macroblock Ordering (FMO)".

5. Attached hereto as **Attachment 4** is a true and correct copy of relevant excerpts of the ITU-T H.264 Standard showing "constraint_set2_flag equal to 1 indicates that the bitstream obeys all constraints specified in subclause A.2.3" and subclause "A.2.3 Extended profile" constraints found at https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-200305-S!!PDF-E&type=items (last accessed on November 19, 2019).

6. Attached hereto as **Attachment 5** is a true and correct copy of a Google forum H264 Codec Compatibility post showing that "Profile: 42e01f" has "Constraint_set2_flag:1" found at https://groups.google.com/forum/#!topic/discuss-webrtc/JGNaniOg688 (last accessed on November 20, 2019).

7. Attached hereto as **Attachment 6** is a true and correct copy of Google public source code for WebRTC showing support for profile 42e01f found at https://chromium.googlesource.com/external/webrtc/+/HEAD/common_video/h264/profile_level_id_unittest.cc (last accessed on November 20, 2019).

8. Attached hereto as **Attachment 7** is a true and correct copy of an article titled *Google Engineers Detail tech Behind Stadia's "Imperceptible" Latency* quoting a Google engineer "So on the receiver side, we use WebRTC extensions provided by our team in Sweden to disable buffering and display things as soon as they arrive" found at https://www.pcgamesn.com/stadia/google-game-streaming-technology-latency-quality (last accessed on November 20, 2019).

9. Attached hereto as **Attachment 8** is a true and correct copy of printout from source.android.com/devices/media (last accessed on November 20, 2019) stating "Android includes Stagefright, a media playback engine at the native level that has built-in software-based codecs for popular media formats" and "Stagefright audio and video playback features include integration with OpenMAX codecs."

10. Attached hereto as **Attachment 9** is a true and correct copy of Android Stagefright source code showing use of FMO "picParam->slice_group_map_type = encParam->fmo_type; switch (encParam->fmo_type) found at https://android.googlesource.com/platform/frameworks/av/+/050ff19e650a53fd2b6f1f2490758b4bf3104ca4/media/libstagefright/codecs/avc/enc/src/init.cpp (last accessed on November 20, 2019).

11. Attached hereto as **Attachment 10** is a true and correct copy of Android Stagefright OpenMAX codec source code showing use of FMO "OMX_BOOL bEnableFMO" found at https://android.googlesource.com/platform/frameworks/av/+/bbba88c/include/media/stagefright/openmax/OMX_Video.h (last accessed on November 20, 2019).

12. Attached hereto as **Attachment 11** is a true and correct copy of publicly available Google Chrome source code for WebRTC suggesting use of SP/SI frames as kSp and kSi frames "enum SliceType : uint8_t { kP = 0, kB = 1, kI = 2, kSp = 3, kSi = 4 }" found at

https://chromium.googlesource.com/external/webrtc/+/refs/heads/master/common_video/h264/h264_common.h (last accessed on November 20, 2019.)

13. Attached hereto as **Attachment 12** is a true and correct copy of relevant excerpts from Microsoft protocols "[MS-RTVPF]: RTP Payload Format for RT Video Streams Extensions" found at https://docs.microsoft.com/en-us/openspecs/office_protocols/ms-rtvpf (last accessed November 19, 2019), including: 1 Introduction noting that the [MS-RTVPF] protocol "describe[es] the payload format for carrying real-time video streams in the payload of the real-Time Transport Protocol"; 1.1 Glossary defining "Super P-frame (SP-frame)"; 3.1.5.5 SP-Frame and Cached Frame Mechanisms, noting "The decoder on the receiver side MUST cache the cached frame because the next SP-frame references it"; "In this case, the P-frame is called a Super P-frame (SP-frame) because it is predicted from the latest cached frame and not from the previous P-frame or the previous I-frame. The presence of an SP-frame in the packetized video bitstream is signaled by the SP field in the Packet Payload Format, as described in sections 2.2.2, 2.2.3, and 2.2.4. Upon receiving an SP-frame, the decoder decodes the video frame using the cached frame reference"; 2.2.2 RTVideo Basic RTP Payload Format, 2.2.3 RTVideo RTP Extended Payload Format, and 2.2.4 RTVideo RTP Extended 2 Payload Format, each showing use of SP-frames "C - SP (1 bit): Super P-frame (SP-frame) flag. A value of one specifies an SPframe. A value of zero specifies the frame is not an SP-frame."

14. Attached hereto as **Attachments 13, 14, and 15** are true and correct copy of correspondence between the parties concerning deficient productions.

15. Attached hereto as **Attachments 16** is [redacted]

[redacted].

16. Attached hereto as **Attachments 17** is [redacted].

<u>Uniloc Needs Additional Discovery</u>

17. There has not been substantive technical discovery in these cases to date, nor has there been a claim construction order. The discovery cut-off data and *Markman* hearing dates are as follows:

	Markman Hearing	Discovery Cutoff
2:18-cv-00497	January 6, 2019	March 30, 2020
2:18-cv-00501	January 7, 2019	March 30, 2020
2:18-cv-00551	February 20, 2020	March 30, 2020

18. As demonstrated by the documents described above and in Uniloc's Opposition to Google's Motion for Summary Judgment of Non-Infringement, as well as in Uniloc's Infringement Contentions, Uniloc has uncovered publicly available evidence disputing Google's assertion that Google does not use SP/SI frames or Flexible Macroblock Ordering ("FMO") and has never implemented SP/SI frames or FMO. Moreover, the publicly available evidence suggests that, despite Google's assertions to the contrary, these techniques are implemented in certain of the accused products, including Stadia accused in the <u>-497 case</u> and Android devices accused in the <u>-501 case</u>.

19. The publicly available evidence also suggests that Google may be using SP/SI frames but calling them kSp and kSi frames.

20. Uniloc needs to conduct its technical discovery into the structure, function, and operation of the accused devices in order to fully respond to Google's premature Motion for Summary Judgment of Non-Infringement.

How Additional Discovery Will Likely Create a Genuine Issue of Material Fact

21. Uniloc's technical discovery into whether and how the accused products partially decode coded data so as to obtain blocks of prediction-error pixels, modify the blocks of prediction-error pixels so as to obtain modified partially decoded data, and complementary code the modified partially decoded data, so as to obtain coded modified data as required by claim 1 of the '934 Patent or have a data-modifying assembly that uses a partial decoder, data modifier, and complementary coder to achieve that result as required by claim 3 of the '934 Patent will likely create a genuine issue of material fact use in the -497 case. Uniloc's Infringement Contentions allege that the accused products accomplish this through use of SP/SI frames. Google claims it does not use SP/SI frames but does not state that it does not partially decode coded data so as to obtain blocks of prediction-error pixels, modify the blocks of prediction-error pixels so as to obtain modified partially decoded data, and complementary code the modified partially decoded data, so as to obtain coded modified data. Uniloc believes that discovery will show that the accused products partially decode coded data so as to obtain blocks of prediction-error pixels, modify the blocks of prediction-error pixels so as to obtain modified partially decoded data, and complementary code the modified partially decoded data, so as to obtain coded modified data either through SP/SI frames or other techniques. For example, public evidence exists of Google's use of kSp and kSi slices or frames in Google's implementation of WebRTC, including in accused product Stadia. Discovery will create a question of fact as to whether this technique is the same as using SP/SI frames and/or the method steps and devices claimed in the Asserted Patents. Uniloc should be afforded the opportunity to

take discovery of Google on the use of kSp and kSi slices or frames relative to the method steps and devices claimed in the Asserted Patents. As such, this discovery will likely create a genuine issue of material fact.

22. Uniloc's technical discovery into whether and how the accused products have a "video encoder for processing a sequence of animated pictures, said encoder comprising": "means for dividing a screen window occupied by said sequence into X rows and Y columns"; "means for separately encoding each one of the X·Y parts of each picture of the sequence thus obtained"; and "means for associating, to each of said parts, a specific label indicating a position of the part in the window, and for encoding these labels in a random order" as required by claim 1 of the '515 Patent will likely create a genuine issue of material fact in the -501 case. Uniloc's Infringement Contentions allege that the accused products accomplish this through use of FMO. Google claims it does not use FMO for certain products but does not state it does not have a video encoder comprised of the elements as claimed in claim 1 of the '515 Patent. Technical discovery into the video encoders in the accused products will likely create a dispute of fact. For example, publicly available evidence suggests that, despite Google's assertions to the contrary, Google uses of FMO within the Android operating system's Stagefright media playback engine that has built-in software-based codecs. The accused Pixel devices have the Android operating system. Discovery will create a question of fact as to whether FMO is used in the accused devices. Uniloc should be afforded the opportunity to take discovery of Google on the use of FMO within the Android operating system, including in the accused Google Pixel devices. As such, this discovery will likely create a genuine issue of material fact.

23. Uniloc's technical discovery into whether and how the accused products transcode a primary encoded signal comprising a sequence of pictures into a secondary encoded signal using

the method steps claimed in claims 1, 4, and 5 of the '960 Patent will likely create a genuine issue of material fact in the -551 case. Uniloc cited to SP/SI frames for support in its infringement contentions but SP/SI frames are not claimed or required. Google's declarations simply state that the accused products do not use SP/SI frames. Google offers no evidence that it does not perform the steps claimed in claims 1, 4, and 5 of the '960 Patent. Technical discovery into the transcoding performed by the accused products will likely create a genuine issue of material fact.

I declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct.

Executed this 13th day of December, 2019 in Collin County, Texas.

/s/ Ryan S. Loveless
Ryan S. Loveless
TX Bar No. 24036997
Etheridge Law Group, PLLC
2600 E. Southlake Blvd., Suite 120 / 324
Southlake, TX 76092
Tel.: (817) 470-7249
Jim@EtheridgeLaw.com
Ryan@EtheridgeLaw.com

Attorneys for Plaintiff Uniloc 2017, LLC

ATTACHMENT 1

- Home
- FAQ
- Source Code
- Issue Tracker

Follow us on

Home

Cisco has taken their H.264 implementation, and open sourced it under BSD license terms. Development and maintenance will be overseen by a board from industry and the open source community. Furthermore, we have provided a binary form suitable for inclusion in applications across a number of different operating systems, and make this binary module available for download from the Internet. We will not pass on our MPEG-LA licensing costs for this module, and based on the current licensing environment, this will effectively make H.264 free for use on supported platforms.

More information at Cisco blog post at:
http://blogs.cisco.com/collaboration/open-source-h-264-removes-barriers-webrtc .

And the Mozilla blog post at:
https://blog.mozilla.org/blog/2013/10/30/video-interoperability-on-the-web-gets-a-boost-from-ciscos-h-264-codec

Source Code https://github.com/cisco/openh264	Issue Tracker https://github.com/cisco/openh264/issues

Home
|
FAQ
|
Source Code
|
Issue Tracker

ATTACHMENT 2

Chrome Platform Status

All features | Releases | Samples | Stats

H.264 software encoder/decoder in Chrome for WebRTC

Realtime / Communication

Include a H.264 video codec encoder and decoder in Chrome for use with WebRTC. At IETF in late november 2014, a compromise was reached with the main contributors to WebRTC to ship both VP8 and H.264. This launch is to follow up in this public commitment. The plan is to use the OpenH264 (same lib as Firefox uses) for encoding and FFmpeg (which is already used elsewhere in Chrome) for decoding.

Specification

Editor's draft

Status in Chromium

Blink components: Blink

Enabled by default (tracking bug) in:
Chrome for desktop release 52
Chrome for Android release 52
Android WebView release 52

Consensus & Standardization

After a feature ships in Chrome, the values listed here are not guaranteed to be up to date.

Firefox:	Shipped
Edge:	Public support
Safari:	No public signals
Web Developers:	No signals

Owners

hbos@chromium.org
torbjorng@chromium.org
hta@chromium.org
tommi@chromium.org

Last updated on 2017-06-14

ATTACHMENT 3

Google Git Sign in

chromium / external / webrtc / fb11424551dae924869ae54059cb1612836cb6f7 / . / webrtc / build / **webrtc.gni**

blob: 28a9ef52d3b25efc44d30a974518ae52bc07a1ce [file] [log] [blame]

```
# Copyright (c) 2014 The WebRTC project authors. All Rights Reserved.
#
# Use of this source code is governed by a BSD-style license
# that can be found in the LICENSE file in the root of the source
# tree. An additional intellectual property rights grant can be found
# in the file PATENTS.  All contributing project authors may
# be found in the AUTHORS file in the root of the source tree.

import("//build/config/arm.gni")
import("//build/config/features.gni")
import("//build/config/mips.gni")
import("//build_overrides/webrtc.gni")

declare_args() {
  # Disable this to avoid building the Opus audio codec.
  rtc_include_opus = true

  # Disable to use absolute header paths for some libraries.
  rtc_relative_path = true

  # Used to specify an external Jsoncpp include path when not compiling the
  # library that comes with WebRTC (i.e. rtc_build_json == 0).
  rtc_jsoncpp_root = "//third_party/jsoncpp/source/include"

  # Used to specify an external OpenSSL include path when not compiling the
  # library that comes with WebRTC (i.e. rtc_build_ssl == 0).
  rtc_ssl_root = ""

  # Selects fixed-point code where possible.
  rtc_prefer_fixed_point = false

  # Enable data logging. Produces text files with data logged within engines
  # which can be easily parsed for offline processing.
  rtc_enable_data_logging = false
```

```
    rtc_prefer_fixed_point = true
  }

  if (!is_ios && (current_cpu != "arm" || arm_version >= 7) &&
      current_cpu != "mips64el") {
    rtc_use_openmax_dl = true
  } else {
    rtc_use_openmax_dl = false
  }

  # Determines whether NEON code will be built.
  rtc_build_with_neon =
      (current_cpu == "arm" && arm_use_neon) || current_cpu == "arm64"

  # Enable this to use HW H.264 encoder/decoder on iOS PeerConnections.
  # Enabling this may break interop with Android clients that support H264.
  rtc_use_objc_h264 = false

  # Enable this to build OpenH264 encoder/FFmpeg decoder. This is supported on
  # all platforms except Android and iOS. Because FFmpeg can be built
  # with/without H.264 support, |ffmpeg_branding| has to separately be set to a
  # value that includes H.264, for example "Chrome". If FFmpeg is built without
  # H.264, compilation succeeds but |H264DecoderImpl| fails to initialize. See
  # also: |rtc_initialize_ffmpeg|.
  # CHECK THE OPENH264, FFMPEG AND H.264 LICENSES/PATENTS BEFORE BUILDING.
  # http://www.openh264.org, https://www.ffmpeg.org/
  rtc_use_h264 = proprietary_codecs && !is_android && !is_ios

  # Determines whether QUIC code will be built.
  rtc_use_quic = false

  # FFmpeg must be initialized for |H264DecoderImpl| to work. This can be done
  # by WebRTC during |H264DecoderImpl::InitDecode| or externally. FFmpeg must
  # only be initialized once. Projects that initialize FFmpeg externally, such
  # as Chromium, must turn this flag off so that WebRTC does not also
  # initialize.
  rtc_initialize_ffmpeg = !build_with_chromium

  # Build sources requiring GTK. NOTICE: This is not present in Chrome OS
  # build environments, even if available for Chromium builds.
  rtc_use_gtk = !build_with_chromium
}

```

Google Git Sign in

chromium / external / github.com / cisco / openh264 / 4c74cc8fdeba12f8556528aca6f12352331cab67 / . / codec / decoder / core / inc / **fmo.h**

blob: 2dc11496a9012c91837b8b18ce88fb32f95b771b [file] [log] [blame]

```
/*!
 * \copy
 *     Copyright (c)  2009-2013, Cisco Systems
 *     All rights reserved.
 *
 *     Redistribution and use in source and binary forms, with or without
 *     modification, are permitted provided that the following conditions
 *     are met:
 *
 *        * Redistributions of source code must retain the above copyright
 *          notice, this list of conditions and the following disclaimer.
 *
 *        * Redistributions in binary form must reproduce the above copyright
 *          notice, this list of conditions and the following disclaimer in
 *          the documentation and/or other materials provided with the
 *          distribution.
 *
 *     THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 *     "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 *     LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
 *     FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
 *     COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
 *     INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
 *     BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 *     LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
 *     CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 *     LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
 *     ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 *     POSSIBILITY OF SUCH DAMAGE.
 *
 *
 * \file    fmo.h
 *
 * \brief   Flexible Macroblock Ordering implementation
 *
 * \date    2/4/2009 Created
 *
 *************************************************************************************
 */
#ifndef WELS_FLEXIBLE_MACROBLOCK_ORDERING_H__
#define WELS_FLEXIBLE_MACROBLOCK_ORDERING_H__

#include "typedefs.h"
#include "wels_const.h"
#include "parameter_sets.h"
#include "memory_align.h"

namespace WelsDec {

#ifndef MB_XY_T
```

```

#define MB_XY_T int32_t
#endif//MB_XY_T

/*!
 * \brief   Wels Flexible Macroblock Ordering (FMO)
 */
typedef struct TagFmo {
uint8_t*        pMbAllocMap;
int32_t         iCountMbNum;
int32_t         iSliceGroupCount;
int32_t         iSliceGroupType;
bool            bActiveFlag;
uint8_t         uiReserved[3];          // reserved padding bytes
} SFmo, *PFmo;


/*!
 * \brief   Initialize Wels Flexible Macroblock Ordering (FMO)
 *
 * \param   pFmo        Wels fmo to be initialized
 * \param   pPps        PPps
 * \param   kiMbWidth   mb width
 * \param   kiMbHeight  mb height
 *
 * \return  0 - successful; none 0 - failed;
 */
int32_t InitFmo (PFmo pFmo, PPps pPps, const int32_t kiMbWidth, const int32_t kiMbHeight, CMemoryAlign* pMa);

/*!
 * \brief   Uninitialize Wels Flexible Macroblock Ordering (FMO) list
 *
 * \param   pFmo        Wels base fmo ptr to be uninitialized
 * \param   kiCnt       count number of PPS per list
 * \param   kiAvail     count available number of PPS in list
 *
 * \return  NONE
 */
void UninitFmoList (PFmo pFmo, const int32_t kiCnt, const int32_t kiAvail, CMemoryAlign* pMa);

/*!
 * \brief   update/insert FMO parameter unit
 *
 * \param   pFmo    FMO context
 * \param   pSps    PSps
 * \param   pPps    PPps
 * \param   pActiveFmoNum   int32_t* [in/out]
 *
 * \return  true - update/insert successfully; false - failed;
 */
int32_t FmoParamUpdate (PFmo pFmo, PSps pSps, PPps pPps, int32_t* pActiveFmoNum, CMemoryAlign* pMa);

/*!
 * \brief   Get successive mb to be processed with given current mb_xy
 *
 * \param   pFmo            Wels fmo context
 * \param   iMbXy           current mb_xy
 *
 * \return  iNextMb - successful; -1 - failed;
 */
MB_XY_T FmoNextMb (PFmo pFmo, const MB_XY_T kiMbXy);

} // namespace WelsDec

#endif//WELS_FLEXIBLE_MACROBLOCK_ORDERING_H__
```

Powered by Gitiles| Privacy txt json

ATTACHMENT 4

INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

H.264
(05/2003)

SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS
Infrastructure of audiovisual services – Coding of moving video

Advanced video coding for generic audiovisual services

ITU-T Recommendation H.264

constraint_set0_flag equal to 1 indicates that the bitstream obeys all constraints specified in subclause A.2.1. constraint_set0_flag equal to 0 indicates that the bitstream may or may not obey all constraints specified in subclause A.2.1.

constraint_set1_flag equal to 1 indicates that the bitstream obeys all constraints specified in subclause A.2.2. constraint_set1_flag equal to 0 indicates that the bitstream may or may not obey all constraints specified in subclause A.2.2.

constraint_set2_flag equal to 1 indicates that the bitstream obeys all constraints specified in subclause A.2.3. constraint_set2_flag equal to 0 indicates that the bitstream may or may not obey all constraints specified in subclause A.2.3.

NOTE – When more than one of constraint_set0_flag, constraint_set1_flag, or constraint_set2_flag are equal to 1, the bitstream obeys the constraints of all of the indicated subclauses of subclause A.2.

reserved_zero_5bits shall be equal to 0 in bitstreams conforming to this Recommendation | International Standard. Other values of reserved_zero_5bits may be specified in the future by ITU-T | ISO/IEC. Decoders shall ignore the value of reserved_zero_5bits.

seq_parameter_set_id identifies the sequence parameter set that is referred to by the picture parameter set. The value of seq_parameter_set_id shall be in the range of 0 to 31, inclusive.

NOTE – When feasible, encoders should use distinct values of seq_parameter_set_id when the values of other sequence parameter set syntax elements differ rather than changing the values of the syntax elements associated with a specific value of seq_parameter_set_id.

log2_max_frame_num_minus4 specifies the value of the variable MaxFrameNum that is used in frame_num related derivations as follows:

$$\text{MaxFrameNum} = 2^{(\text{ log2\_max\_frame\_num\_minus4} + 4\text{ })} \quad (7\text{-}1)$$

The value of log2_max_frame_num_minus4 shall be in the range of 0 to 12, inclusive.

pic_order_cnt_type specifies the method to decode picture order count (as specified in subclause 8.2.1). The value of pic_order_cnt_type shall be in the range of 0 to 2, inclusive.

pic_order_cnt_type shall not be equal to 2 in a coded video sequence that contains any of the following

- an access unit containing a non-reference frame followed immediately by an access unit containing a non-reference picture
- two access units each containing a field with the two fields together forming a complementary non-reference field pair followed immediately by an access unit containing a non-reference picture
- an access unit containing a non-reference field followed immediately by an access unit containing another non-reference picture that does not form a complementary non-reference field pair with the first of the two access units

log2_max_pic_order_cnt_lsb_minus4 specifies the value of the variable MaxPicOrderCntLsb that is used in the decoding process for picture order count as specified in subclause 8.2.1 as follows:

$$\text{MaxPicOrderCntLsb} = 2^{(\text{ log2\_max\_pic\_order\_cnt\_lsb\_minus4} + 4\text{ })} \quad (7\text{-}2)$$

The value of log2_max_pic_order_cnt_lsb_minus4 shall be in the range of 0 to 12, inclusive.

delta_pic_order_always_zero_flag equal to 1 specifies that delta_pic_order_cnt[0] and delta_pic_order_cnt[1] are not present in the slice headers of the sequence and shall be inferred to be equal to 0. delta_pic_order_always_zero_flag equal to 0 specifies that delta_pic_order_cnt[0] is present in the slice headers of the sequence and delta_pic_order_cnt[1] may be present in the slice headers of the sequence.

offset_for_non_ref_pic is used to calculate the picture order count of a non-reference picture as specified in 8.2.1. The value of offset_for_non_ref_pic shall be in the range of -2^{31} to $2^{31} - 1$, inclusive.

offset_for_top_to_bottom_field is used to calculate the picture order count of the bottom field in a frame as specified in 8.2.1. The value of offset_for_top_to_bottom_field shall be in the range of -2^{31} to $2^{31} - 1$, inclusive.

num_ref_frames_in_pic_order_cnt_cycle is used in the decoding process for picture order count as specified in subclause 8.2.1. The value of num_ref_frames_in_pic_order_cnt_cycle shall be in the range of 0 to 255, inclusive.

offset_for_ref_frame[i] is an element of a list of num_ref_frames_in_pic_order_cnt_cycle values used in the decoding process for picture order count as specified in subclause 8.2.1. The value of offset_for_ref_frame[i] shall be in the range of -2^{31} to $2^{31} - 1$, inclusive.

– The level constraints specified for the Main profile in subclause A.3 shall be fulfilled.

Conformance of a bitstream to the Main profile is specified by profile_idc being equal to 77.

Decoders conforming to the Main profile at a specified level shall be capable of decoding all bitstreams in which profile_idc is equal to 77 or constraint_set1_flag is equal to 1 and in which level_idc represents a level less than or equal to the specified level.

A.2.3 Extended profile

Bitstreams conforming to the Extended profile shall obey the following constraints:

– Sequence parameter sets shall have direct_8x8_inference_flag equal to 1.

– Picture parameter sets shall have entropy_coding_mode_flag equal to 0.

– Picture parameter sets shall have num_slice_groups_minus1 in the range of 0 to 7, inclusive.

– The level constraints specified for the Extended profile in subclause A.3 shall be fulfilled.

Conformance of a bitstream to the Extended profile is specified by profile_idc being equal to 88.

Decoders conforming to the Extended profile at a specified level shall be capable of decoding all bitstreams in which profile_idc is equal to 88 or constraint_set2_flag is equal to 1 and in which level_idc represents a level less than or equal to specified level.

Decoders conforming to the Extended profile at a specified level shall also be capable of decoding all bitstreams in which profile_idc is equal to 66 or constraint_set0_flag is equal to 1, in which level_idc represents a level less than or equal to the specified level.

A.3 Levels

The following is specified for expressing the constraints in this Annex.

- Let access unit n be the n-th access unit in decoding order with the first access unit being access unit 0.

- Let picture n be the primary coded picture or the corresponding decoded picture of access unit n.

A.3.1 Profile-independent level limits

Let the variable fR be derived as follows.

- If picture n is a frame, fR is set equal to 1 ÷ 172.

- Otherwise (picture n is a field), fR is set equal to 1 ÷ (172 * 2).

Bitstreams conforming to any profile at a specified level shall obey the following constraints:

a) The nominal removal time of access unit n (with n > 0) from the CPB as specified in subclause C.1.2, satisfies the constraint that $t_{r,n}(n) - t_r(n - 1)$ is greater than or equal to Max(PicSizeInMbs ÷ MaxMBPS, fR), where MaxMBPS is the value specified in Table A-1 that applies to picture n, and PicSizeInMbs is the number of macroblocks in picture n.

b) The difference between consecutive output times of pictures from the DPB as specified in subclause C.2.2, satisfies the constraint that $\Delta t_{o,dpb}(n)$ >= Max(PicSizeInMbs ÷ MaxMBPS, fR), where MaxMBPS is the value specified in Table A-1 for picture n, and PicSizeInMbs is the number of macroblocks of picture n, provided that picture n is a picture that is output and is not the last picture of the bitstream that is output.

c) The sum of the NumBytesInNALunit variables for access unit 0 is less than or equal to 256 * ChromaFormatFactor * (PicSizeInMbs + MaxMBPS * ($t_r(0) - t_{r,n}(0)$)) ÷ MinCR, where MaxMBPS and MinCR are the values specified in Table A-1 that apply to picture 0 and PicSizeInMbs is the number of macroblocks in picture 0.

d) The sum of the NumBytesInNALunit variables for access unit n (with n > 0) is less than or equal to 256 * ChromaFormatFactor * MaxMBPS * ($t_r(n) - t_r(n - 1)$) ÷ MinCR, where MaxMBPS and MinCR are the values specified in Table A-1 that apply to picture n.

e) PicWidthInMbs * FrameHeightInMbs <= MaxFS, where MaxFS is specified in Table A-1

f) PicWidthInMbs <= Sqrt(MaxFS * 8)

g) FrameHeightInMbs <= Sqrt(MaxFS * 8)

ATTACHMENT 5

Search for messages

Sign in

Groups

Home

Click on a group's star icon to add it to your favorites

Recently viewed

discuss-webrtc

Privacy - Terms of Service

discuss-webrtc ›

H264 Codec Compatibility

1 post by 1 author

Javier Morgade 8/27/18

Hi guys,

I hope this is the right group for the topic ... I have a question regarding H264 codec compatibility with Webrtc enabled browsers. I hope someone can help.

I have been playing around the Janus-GW streaming plugin and basically we can manage to playback any h264 stream encoded with the following profile_id (42e01f). What basically means baseline profile (66) constraint flag set at (0,1,2). I tried to dig in to the mailing list and happens to be the profille_id mostly tested by other group members.

```
Profile: 42e01f

          0100 0010 = Profile_idc: Baseline profile (66)

          1... .... = Constraint_set0_flag:1

          .1.. .... = Constraint_set1_flag:1

          ..1. .... = Constraint_set2_flag:1

          ...0 .... = Constraint_set3_flag:0

          .... 0000 = Reserved_zero_4bits: 0

          0001 1111 = Level_id: 31 [Level 3.1]
```

On the other hand we cannot manage to get our streams running if we, for instance, run a profile_id (42801e), what also means baseline but with unconstrained flags (1,2).

```
Profile: 42801e

          0100 0010 = Profile_idc: Baseline profile (66)

          1... .... = Constraint_set0_flag:1

          .0.. .... = Constraint_set1_flag:0

          ..0. .... = Constraint_set2_flag:0

          ...0 .... = Constraint_set3_flag:0

          .... 0000 = Reserved_zero_4bits: 0

          0001 1110 = Level_id: 30 [Level 3.0]
```

I was wondering if someone can someone confirm if any h264 baseline (66)

ATTACHMENT 6

Google Git Sign in

chromium / external / webrtc / HEAD / . / common_video / h264 /
profile_level_id_unittest.cc

blob: 957b434a3cbebc1fd12bd7fb719f441f0fd6838a [file] [log] [blame]

```
/*
 *  Copyright (c) 2016 The WebRTC project authors. All Rights Reserved.
 *
 *  Use of this source code is governed by a BSD-style license
 *  that can be found in the LICENSE file in the root of the source
 *  tree. An additional intellectual property rights grant can be found
 *  in the file PATENTS.  All contributing project authors may
 *  be found in the AUTHORS file in the root of the source tree.
 */

#include "common_video/h264/profile_level_id.h"

#include <map>
#include <string>

#include "absl/types/optional.h"
#include "media/base/h264_profile_level_id.h"
#include "test/gtest.h"

namespace webrtc {
namespace H264 {

TEST(H264ProfileLevelId, TestParsingInvalid) {
  // Malformed strings.
  EXPECT_FALSE(ParseProfileLevelId(""));
  EXPECT_FALSE(ParseProfileLevelId(" 42e01f"));
  EXPECT_FALSE(ParseProfileLevelId("4242e01f"));
  EXPECT_FALSE(ParseProfileLevelId("e01f"));
  EXPECT_FALSE(ParseProfileLevelId("gggggg"));

  // Invalid level.
  EXPECT_FALSE(ParseProfileLevelId("42e000"));
  EXPECT_FALSE(ParseProfileLevelId("42e00f"));
  EXPECT_FALSE(ParseProfileLevelId("42e0ff"));

  // Invalid profile.
  EXPECT_FALSE(ParseProfileLevelId("42e11f"));
  EXPECT_FALSE(ParseProfileLevelId("58601f"));
  EXPECT_FALSE(ParseProfileLevelId("64e01f"));
}

TEST(H264ProfileLevelId, TestParsingLevel) {
  EXPECT_EQ(kLevel3_1, ParseProfileLevelId("42e01f")->level);
  EXPECT_EQ(kLevel1_1, ParseProfileLevelId("42e00b")->level);
  EXPECT_EQ(kLevel1_b, ParseProfileLevelId("42f00b")->level);
  EXPECT_EQ(kLevel4_2, ParseProfileLevelId("42C02A")->level);
  EXPECT_EQ(kLevel5_2, ParseProfileLevelId("640c34")->level);
}

TEST(H264ProfileLevelId, TestParsingConstrainedBaseline) {
  EXPECT_EQ(kProfileConstrainedBaseline,
            ParseProfileLevelId("42e01f")->profile);
  EXPECT_EQ(kProfileConstrainedBaseline,
            ParseProfileLevelId("42C02A")->profile);
  EXPECT_EQ(kProfileConstrainedBaseline,
            ParseProfileLevelId("4de01f")->profile);
  EXPECT_EQ(kProfileConstrainedBaseline,
            ParseProfileLevelId("58f01f")->profile);
}

TEST(H264ProfileLevelId, TestParsingBaseline) {
  EXPECT_EQ(kProfileBaseline, ParseProfileLevelId("42a01f")->profile);
  EXPECT_EQ(kProfileBaseline, ParseProfileLevelId("58A01F")->profile);
}

TEST(H264ProfileLevelId, TestParsingMain) {
  EXPECT_EQ(kProfileMain, ParseProfileLevelId("4D401f")->profile);
}

TEST(H264ProfileLevelId, TestParsingHigh) {
  EXPECT_EQ(kProfileHigh, ParseProfileLevelId("64001f")->profile);
}

TEST(H264ProfileLevelId, TestParsingConstrainedHigh) {
  EXPECT_EQ(kProfileConstrainedHigh, ParseProfileLevelId("640c1f")->profile);
}

TEST(H264ProfileLevelId, TestSupportedLevel) {
  EXPECT_EQ(kLevel2_1, *SupportedLevel(640 * 480, 25));
  EXPECT_EQ(kLevel3_1, *SupportedLevel(1280 * 720, 30));
  EXPECT_EQ(kLevel4_2, *SupportedLevel(1920 * 1280, 60));
}

// Test supported level below level 1 requirements.
TEST(H264ProfileLevelId, TestSupportedLevelInvalid) {
  EXPECT_FALSE(SupportedLevel(0, 0));
  // All levels support fps > 5.
  EXPECT_FALSE(SupportedLevel(1280 * 720, 5));
  // All levels support frame sizes > 183 * 137.
  EXPECT_FALSE(SupportedLevel(183 * 137, 30));
}

TEST(H264ProfileLevelId, TestToString) {
  EXPECT_EQ("42e01f", *ProfileLevelIdToString(ProfileLevelId(
                          kProfileConstrainedBaseline, kLevel3_1)));
  EXPECT_EQ("42000a",
            *ProfileLevelIdToString(ProfileLevelId(kProfileBaseline, kLevel1)));
  EXPECT_EQ("4d001f",
            ProfileLevelIdToString(ProfileLevelId(kProfileMain, kLevel3_1)));
  EXPECT_EQ("640c2a", *ProfileLevelIdToString(
                          ProfileLevelId(kProfileConstrainedHigh, kLevel4_2)));
  EXPECT_EQ("64002a",
            *ProfileLevelIdToString(ProfileLevelId(kProfileHigh, kLevel4_2)));
}

TEST(H264ProfileLevelId, TestToStringLevel1b) {
  EXPECT_EQ("42f00b", *ProfileLevelIdToString(ProfileLevelId(
                          kProfileConstrainedBaseline, kLevel1_b)));
  EXPECT_EQ("42100b", *ProfileLevelIdToString(
                          ProfileLevelId(kProfileBaseline, kLevel1_b)));
  EXPECT_EQ("4d100b",
            *ProfileLevelIdToString(ProfileLevelId(kProfileMain, kLevel1_b)));
}

TEST(H264ProfileLevelId, TestToStringRoundTrip) {
  EXPECT_EQ("42e01f", *ProfileLevelIdToString(*ParseProfileLevelId("42e01f")));
  EXPECT_EQ("42e01f", *ProfileLevelIdToString(*ParseProfileLevelId("42E01F")));
  EXPECT_EQ("4d100b", *ProfileLevelIdToString(*ParseProfileLevelId("4d100b")));
  EXPECT_EQ("4d100b", *ProfileLevelIdToString(*ParseProfileLevelId("4D100B")));
  EXPECT_EQ("640c2a", *ProfileLevelIdToString(*ParseProfileLevelId("640c2a")));
  EXPECT_EQ("640c2a", *ProfileLevelIdToString(*ParseProfileLevelId("640C2A")));
}

```

```
  high_level["profile-level-id"] = "42e01f";

  // Level asymmetry is not allowed; test that answer level is the lower of the
  // local and remote levels.
  CodecParameterMap answer_params;
  GenerateProfileLevelIdForAnswer(low_level /* local_supported */,
                                  high_level /* remote_offered */,
                                  &answer_params);
  EXPECT_EQ("42e015", answer_params["profile-level-id"]);

  CodecParameterMap answer_params2;
  GenerateProfileLevelIdForAnswer(high_level /* local_supported */,
                                  low_level /* remote_offered */,
                                  &answer_params2);
  EXPECT_EQ("42e015", answer_params2["profile-level-id"]);
}

TEST(H264ProfileLevelId,
     TestGenerateProfileLevelIdForAnswerConstrainedBaselineLevelAsymmetry) {
  CodecParameterMap local_params;
  local_params["profile-level-id"] = "42e01f";
  local_params["level-asymmetry-allowed"] = "1";
  CodecParameterMap remote_params;
  remote_params["profile-level-id"] = "42e015";
  remote_params["level-asymmetry-allowed"] = "1";
  CodecParameterMap answer_params;
  GenerateProfileLevelIdForAnswer(local_params, remote_params, &answer_params);
  // When level asymmetry is allowed, we can answer a higher level than what was
  // offered.
  EXPECT_EQ("42e01f", answer_params["profile-level-id"]);
}

}  // namespace H264
}  // namespace webrtc
```

Powered by Gitiles| Privacy txt json

ATTACHMENT 7

Jacob Ridley
195 days ago

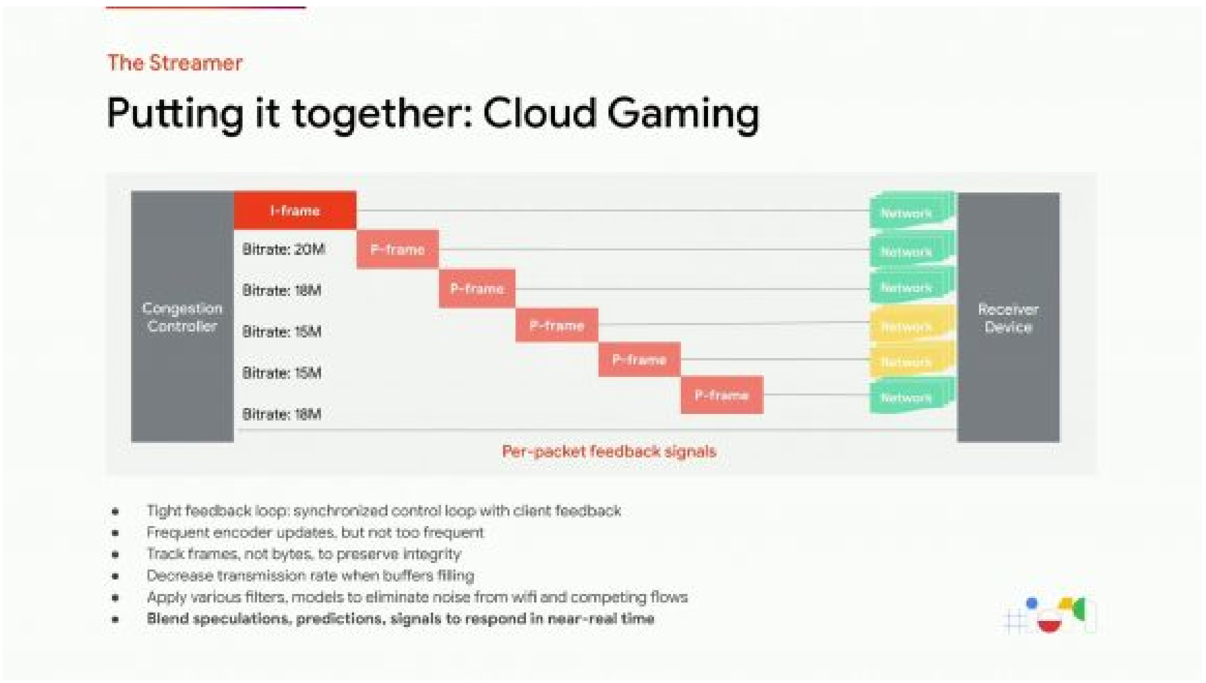
Google engineers detail the streaming tech behind Stadia's "imperceptible" latency

Google has just wrapped up the first day of its annual developer event, Google I/O, which briefly touched on the tech behind Google Stadia. While filling just an hour of the three day event, during that time some of Google's senior staff took to the stage to outline how the company is going about reducing latency to imperceivable levels and

deep, clogging up buffers, and end up causing needlessly massive queues. That queuing alone would usually be a death sentence for game streaming.

"So on the receiver side, we use WebRTC extensions provided by our team in Sweden to disable buffering and display things as soon as they arrive. For our application, controlling congestion to keep the buffering as low as possible is vital."

With an optimised BBR algorithm and a frame-by-frame feedback loop, Google can avoid some of the pitfalls associated with video streaming and laggy web streaming services.


The Streamer

Putting it together: Cloud Gaming

- Tight feedback loop: synchronized control loop with client feedback
- Frequent encoder updates, but not too frequent
- Track frames, not bytes, to preserve integrity
- Decrease transmission rate when buffers filling
- Apply various filters, models to eliminate noise from wifi and competing flows
- **Blend speculations, predictions, signals to respond in near-real time**

"The controller is constantly adjusting a target bitrate based on per packet feedback signals that arrived from the receiver device," McCool continues. "The controller uses a variety of filters, models, and other tools and then synthesises them into a model of the available bandwidth and chooses the target bitrate for the video encoder.

ATTACHMENT 8

Media

Android includes Stagefright, a media playback engine at the native level that has built-in software-based codecs for popular media formats.

Stagefright audio and video playback features include integration with OpenMAX codecs, session management, time-synchronized rendering, transport control, and DRM.

Stagefright also supports integration with custom hardware codecs provided by you. To set a hardware path to encode and decode media, you must implement a hardware-based codec as an OpenMax IL (Integration Layer) component.

Note: Stagefright updates can occur through the Android monthly security update (/security/bulletin/index.html) process and as part of an Android OS release.

Architecture

Media applications interact with the Android native multimedia framework according to the following architecture.

ATTACHMENT 9

Google Git Sign in

android / platform / frameworks / av / 050ff19e650a53fd2b6f1f2490758b4bf3104ca4 / . / media / libstagefright / codecs / avc / enc / src / **init.cpp**

blob: 6e1413ad5007f4880a2f0c2ce896e3d0460a4952 [file] [log] [blame]

```
/* ------------------------------------------------------------------
 * Copyright (C) 1998-2009 PacketVideo
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
 * express or implied.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 * -------------------------------------------------------------------
 */
#include "avcenc_lib.h"
#include "avcenc_api.h"

#define LOG2_MAX_FRAME_NUM_MINUS4   12   /* 12 default */
#define SLICE_GROUP_CHANGE_CYCLE    1    /* default */

/* initialized variables to be used in SPS*/
AVCEnc_Status  SetEncodeParam(AVCHandle* avcHandle, AVCEncParams* encParam,
                              void* extSPS, void* extPPS)
{
    AVCEncObject *encvid = (AVCEncObject*) avcHandle->AVCObject;
    AVCCommonObj *video = encvid->common;
    AVCSeqParamSet *seqParam = video->currSeqParams;
    AVCPicParamSet *picParam = video->currPicParams;
    AVCSliceHeader *sliceHdr = video->sliceHdr;
    AVCRateControl *rateCtrl = encvid->rateCtrl;
    AVCEnc_Status status;
    void *userData = avcHandle->userData;
    int ii, maxFrameNum;

    AVCSeqParamSet* extS = NULL;
    AVCPicParamSet* extP = NULL;

    if (extSPS) extS = (AVCSeqParamSet*) extSPS;
    if (extPPS) extP = (AVCPicParamSet*) extPPS;

    /* This part sets the default values of the encoding options this
    library supports in seqParam, picParam and sliceHdr structures and
    also copy the values from the encParam into the above 3 structures.

    Some parameters will be assigned later when we encode SPS or PPS such as
    the seq_parameter_id or pic_parameter_id. Also some of the slice parameters
    have to be re-assigned per slice basis such as frame_num, slice_type,
    first_mb_in_slice, pic_order_cnt_lsb, slice_qp_delta, slice_group_change_cycle */

    /* profile_idc, constrained_setx_flag and level_idc is set by VerifyProfile(),
    and VerifyLevel() functions later. */

    encvid->fullsearch_enable = encParam->fullsearch;

    encvid->outOfBandParamSet = ((encParam->out_of_band_param_set == AVC_ON) ? TRUE : FALSE);

    /* parameters derived from the the encParam that are used in SPS */
```

```

            return AVCENC_NOT_SUPPORTED;
        }
        seqParam->mb_adaptive_frame_field_flag = extS->mb_adaptive_frame_field_flag;
        if (extS->mb_adaptive_frame_field_flag != FALSE)
        {
            return AVCENC_NOT_SUPPORTED;
        }

        seqParam->direct_8x8_inference_flag = extS->direct_8x8_inference_flag;
        seqParam->frame_cropping_flag = extS->frame_cropping_flag ;
        if (extS->frame_cropping_flag != FALSE)
        {
            return AVCENC_NOT_SUPPORTED;
        }

        seqParam->frame_crop_bottom_offset = 0;
        seqParam->frame_crop_left_offset = 0;
        seqParam->frame_crop_right_offset = 0;
        seqParam->frame_crop_top_offset = 0;
        seqParam->vui_parameters_present_flag = extS->vui_parameters_present_flag;
        if (extS->vui_parameters_present_flag)
        {
            memcpy(&(seqParam->vui_parameters), &(extS->vui_parameters), sizeof(AVCVUIParams));
        }
    }
    else
    {
        return AVCENC_NOT_SUPPORTED;
    }

    /***************** now PPS ******************************/
    if (!extP && !extS)
    {
        picParam->pic_parameter_set_id = (uint)(-1); /* start with zero */
        picParam->seq_parameter_set_id = (uint)(-1); /* start with zero */
        picParam->entropy_coding_mode_flag = 0; /* default to CAVLC */
        picParam->pic_order_present_flag = 0; /* default for now, will need it for B-slice */
        /* FMO */
        if (encParam->num_slice_group < 1 || encParam->num_slice_group > MAX_NUM_SLICE_GROUP)
        {
            return AVCENC_INVALID_NUM_SLICEGROUP;
        }
        picParam->num_slice_groups_minus1 = encParam->num_slice_group - 1;

        if (picParam->num_slice_groups_minus1 > 0)
        {
            picParam->slice_group_map_type = encParam->fmo_type;
            switch (encParam->fmo_type)
            {
                case 0:
                    for (ii = 0; ii <= (int)picParam->num_slice_groups_minus1; ii++)
                    {
                        picParam->run_length_minus1[ii] = encParam->run_length_minus1[ii];
                    }
                    break;
                case 2:
                    for (ii = 0; ii < (int)picParam->num_slice_groups_minus1; ii++)
                    {
                        picParam->top_left[ii] = encParam->top_left[ii];
                        picParam->bottom_right[ii] = encParam->bottom_right[ii];
                    }
                    break;
                case 3:
                case 4:
                case 5:
                    if (encParam->change_dir_flag == AVC_ON)
                    {
                        picParam->slice_group_change_direction_flag = TRUE;
```

ATTACHMENT 10

Google Git Sign in

android / platform / frameworks / av / bbba88c / . / include / media / stagefright / openmax / **OMX_Video.h**

blob: 4f8485d3fa0bc4aeac9ff1f48e47f6bc0848506c [file] [log] [blame]

```
/* -------------------------------------------------------------------
 * Copyright (C) 1998-2009 PacketVideo
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
 * express or implied.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 * -------------------------------------------------------------------
 */
/**
 * Copyright (c) 2008 The Khronos Group Inc.
 *
 * Permission is hereby granted, free of charge, to any person obtaining
 * a copy of this software and associated documentation files (the
 * "Software"), to deal in the Software without restriction, including
 * without limitation the rights to use, copy, modify, merge, publish,
 * distribute, sublicense, and/or sell copies of the Software, and to
 * permit persons to whom the Software is furnished to do so, subject
 * to the following conditions:
 * The above copyright notice and this permission notice shall be included
 * in all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
 * OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
 * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
 * IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
 * CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
 * TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 *
 */

/**
 *  @file OMX_Video.h - OpenMax IL version 1.1.2
 *  The structures is needed by Video components to exchange parameters
 *  and configuration data with OMX components.
 */
#ifndef OMX_Video_h
#define OMX_Video_h

/** @defgroup video OpenMAX IL Video Domain
 * @ingroup iv
 * Structures for OpenMAX IL Video domain
 * @{
 */

#ifdef __cplusplus
extern "C" {
#endif /* __cplusplus */


/**
```

```
 *  bEnableFMO                : Enable/disable flexible macroblock ordering
 *  bEnableASO                : Enable/disable arbitrary slice ordering
 *  bEnableRS                 : Enable/disable sending of redundant slices
 *  eProfile                  : AVC profile(s) to use
 *  eLevel                    : AVC level(s) to use
 *  nAllowedPictureTypes      : Specifies the picture types allowed in the
 *                              bitstream
 *  bFrameMBsOnly             : specifies that every coded picture of the
 *                              coded video sequence is a coded frame
 *                              containing only frame macroblocks
 *  bMBAFF                    : Enable/disable switching between frame and
 *                              field macroblocks within a picture
 *  bEntropyCodingCABAC       : Entropy decoding method to be applied for the
 *                              syntax elements for which two descriptors appear
 *                              in the syntax tables
 *  bWeightedPPrediction      : Enable/disable weighted prediction shall not
 *                              be applied to P and SP slices
 *  nWeightedBipredicitonMode : Default weighted prediction is applied to B
 *                              slices
 *  bconstIpred               : Enable/disable intra prediction
 *  bDirect8x8Inference       : Specifies the method used in the derivation
 *                              process for luma motion vectors for B_Skip,
 *                              B_Direct_16x16 and B_Direct_8x8 as specified
 *                              in subclause 8.4.1.2 of the AVC spec
 *  bDirectSpatialTemporal    : Flag indicating spatial or temporal direct
 *                              mode used in B slice coding (related to
 *                              bDirect8x8Inference) . Spatial direct mode is
 *                              more common and should be the default.
 *  nCabacInitIdx             : Index used to init CABAC contexts
 *  eLoopFilterMode           : Enable/disable loop filter
 */
typedef struct OMX_VIDEO_PARAM_AVCTYPE {
    OMX_U32 nSize;
    OMX_VERSIONTYPE nVersion;
    OMX_U32 nPortIndex;
    OMX_U32 nSliceHeaderSpacing;
    OMX_U32 nPFrames;
    OMX_U32 nBFrames;
    OMX_BOOL bUseHadamard;
    OMX_U32 nRefFrames;
        OMX_U32 nRefIdx10ActiveMinus1;
        OMX_U32 nRefIdx11ActiveMinus1;
    OMX_BOOL bEnableUEP;
    OMX_BOOL bEnableFMO;
    OMX_BOOL bEnableASO;
    OMX_BOOL bEnableRS;
    OMX_VIDEO_AVCPROFILETYPE eProfile;
        OMX_VIDEO_AVCLEVELTYPE eLevel;
    OMX_U32 nAllowedPictureTypes;
        OMX_BOOL bFrameMBsOnly;
    OMX_BOOL bMBAFF;
    OMX_BOOL bEntropyCodingCABAC;
    OMX_BOOL bWeightedPPrediction;
    OMX_U32 nWeightedBipredicitonMode;
    OMX_BOOL bconstIpred ;
    OMX_BOOL bDirect8x8Inference;
        OMX_BOOL bDirectSpatialTemporal;
        OMX_U32 nCabacInitIdc;
        OMX_VIDEO_AVCLOOPFILTERTYPE eLoopFilterMode;
} OMX_VIDEO_PARAM_AVCTYPE;

typedef struct OMX_VIDEO_PARAM_PROFILELEVELTYPE {
   OMX_U32 nSize;
   OMX_VERSIONTYPE nVersion;
   OMX_U32 nPortIndex;
   OMX_U32 eProfile;      /**< type is OMX_VIDEO_AVCPROFILETYPE, OMX_VIDEO_H263PROFILETYPE,
                                 or OMX_VIDEO_MPEG4PROFILETYPE depending on context */
```

ATTACHMENT 11

Google Git Sign in

chromium / external / webrtc / refs/heads/master / . / common_video / h264 /
h264_common.h

blob: 2beef16ac531d100c464a0ed1292f127b511ea79 [file] [log] [blame]

```
/*
 *  Copyright (c) 2016 The WebRTC project authors. All Rights Reserved.
 *
 *  Use of this source code is governed by a BSD-style license
 *  that can be found in the LICENSE file in the root of the source
 *  tree. An additional intellectual property rights grant can be found
 *  in the file PATENTS.  All contributing project authors may
 *  be found in the AUTHORS file in the root of the source tree.
 */

#ifndef COMMON_VIDEO_H264_H264_COMMON_H_
#define COMMON_VIDEO_H264_H264_COMMON_H_

#include <stddef.h>
#include <stdint.h>

#include <vector>

#include "rtc_base/buffer.h"

namespace webrtc {

namespace H264 {
// The size of a full NALU start sequence {0 0 0 1}, used for the first NALU
// of an access unit, and for SPS and PPS blocks.
const size_t kNaluLongStartSequenceSize = 4;

// The size of a shortened NALU start sequence {0 0 1}, that may be used if
// not the first NALU of an access unit or an SPS or PPS block.
const size_t kNaluShortStartSequenceSize = 3;

// The size of the NALU type byte (1).
const size_t kNaluTypeSize = 1;

enum NaluType : uint8_t {

  kSlice = 1,
  kIdr = 5,
  kSei = 6,
  kSps = 7,
  kPps = 8,
  kAud = 9,
  kEndOfSequence = 10,
  kEndOfStream = 11,
  kFiller = 12,
  kStapA = 24,
  kFuA = 28
};

enum SliceType : uint8_t { kP = 0, kB = 1, kI = 2, kSp = 3, kSi = 4 };

struct NaluIndex {
  // Start index of NALU, including start sequence.
  size_t start_offset;
  // Start index of NALU payload, typically type header.
  size_t payload_start_offset;
  // Length of NALU payload, in bytes, counting from payload_start_offset.
  size_t payload_size;
};

// Returns a vector of the NALU indices in the given buffer.
std::vector<NaluIndex> FindNaluIndices(const uint8_t* buffer,
                                       size_t buffer_size);

// Get the NAL type from the header byte immediately following start sequence.
NaluType ParseNaluType(uint8_t data);

// Methods for parsing and writing RBSP. See section 7.4.1 of the H264 spec.
//
// The following sequences are illegal, and need to be escaped when encoding:
// 00 00 00 -> 00 00 03 00
// 00 00 01 -> 00 00 03 01
// 00 00 02 -> 00 00 03 02
// And things in the source that look like the emulation byte pattern (00 00 03)
// need to have an extra emulation byte added, so it's removed when decoding:
// 00 00 03 -> 00 00 03 03
//
// Decoding is simply a matter of finding any 00 00 03 sequence and removing
// the 03 emulation byte.
```

ATTACHMENT 12

1 Introduction

02/14/2019 • 2 minutes to read

The RTP Payload Format for RTVideo Streams Extensions [MS-RTVPF] protocol is a proprietary protocol describing the payload format for carrying real-time video streams in the payload of the Real-Time Transport Protocol (RTP). It is used to transmit and receive real-time video streams in two-party peer-to-peer calls and in multi-party conference calls.

Sections 1.5, 1.8, 1.9, 2, and 3 of this specification are normative. All other sections and examples in this specification are informative.

Is this page helpful?

Yes No

1.1 Glossary

02/14/2019 • 2 minutes to read

This document uses the following terms:

B-frame: A bidirectional video frame that references both the previous frame and the next frame.

big-endian: Multiple-byte values that are byte-ordered with the most significant byte stored in the memory location with the lowest address.

cached frame: A video frame that is cached for later use by an encoder and a decoder. A cached frame acts as a reference frame for the next Super P-frame (SP-frame). I-frames and SP-frames typically are cached frames.

endpoint: A device that is connected to a computer network.

entry point header: A header field whose values specify the horizontal and vertical dimensions of a video frame. See also sequence header.

forward error correction (FEC): A process in which a sender uses redundancy to enable a receiver to recover from packet loss.

GOP: A group of pictures that starts with one I-frame and ends with the next I-frame, excluding the next I-frame, as described in [SMPTE-VC-1].

I-frame: A video frame that is encoded as a single image, such that it can be decoded without any dependencies on previous frames. Also referred to as Intra-Coded frame, Intra frame, and key frame.

maximum transmission unit (MTU): The size, in bytes, of the largest packet that a given layer of a communications protocol can pass onward.

network byte order: The order in which the bytes of a multiple-byte number are transmitted on a network, most significant byte first (in big-endian storage). This may or may not match the order in which numbers are normally stored in memory for a particular processor.

P-frame: A predicative video frame that references a previous frame. Also referred to as inter-coded frame or inter-frame.

Real-Time Transport Protocol (RTP): A network transport protocol that provides end-to-end transport functions that are suitable for applications that transmit real-time data, such as audio and video, as described in [RFC3550].

RTP packet: A data packet consisting of the fixed RTP header, a possibly empty list of contributing sources, and the payload data. Some underlying protocols may require an encapsulation of the RTP packet to be defined. Typically one packet of the underlying protocol contains a single RTP packet, but several RTP packets can be contained if permitted by the encapsulation method. See [RFC3550] section 3.

RTP payload: The data transported by RTP in a packet, for example audio samples or compressed video data. For more information, see [RFC3550] section 3.

RTVC1: A Microsoft proprietary implementation of the VC1 codec for real-time transmission purposes, as described in [SMPTE-VC-1]. Microsoft extensions to VC1 are based on cached frame and SP-frame, as described in [MS-RTVPF].

RTVideo: A video stream that carries an RTVC1 bit stream.

RTVideo FEC metadata packet: A packet that is generated by using the forward error correction (FEC) algorithm to provide redundancy. It is packetized in the RTVideo FEC Real-Time Transport Protocol (RTP) payload format.

RTVideo frame: A video frame that is encoded by using an RTVC1 codec.

sequence header: A set of encoding and display parameters that are placed before a group of pictures, as described in [SMPTE-VC-1]. See also entry point header.

Super P-frame (SP-frame): A special P-frame that uses the previous cached frame instead of the previous P-frame or I-frame as a reference frame.

3.1.5.5 SP-Frame and Cached Frame mechanisms

02/14/2019 • 2 minutes to read

Both encoder and decoder can periodically cache decoded video frames. The cached video frame is stored in a dedicated memory location in addition to the current reference decoded I and P frames. The caching mechanism is done in a synchronized fashion, meaning that the video encoder and the video decoder MUST have a copy of the same decoded video frame in their cache whenever the encoder has finished encoding a video frame and the decoder is about to decode the same video frame.

The decoder on the receiver side MUST cache the cached frame because the next SP-frame references it.

The cache frame is signaled in the packetized video bitstream by means of the **C** field in the Packet Payload Format, as described in sections 2.2.2, 2.2.3 and 2.2.4. The encoder and decoder use only one cached frame at a time, so the data of the previous cached frame can be discarded whenever a new frame is cached.

When the encoder receives a packet loss event reported by the decoder, it can choose to encode the next P-frame using the cached frame as reference, as opposed to the previous P-frame or the previous I-frame. In this case, the P-frame is called a Super P-frame (SP-frame) because it is predicted from the latest cached frame and not from the previous P-frame or the previous I-frame. The presence of an SP-frame in the packetized video bitstream is signaled by the **SP** field in the Packet Payload Format, as described in sections 2.2.2, 2.2.3 and 2.2.4. Upon receiving an SP-frame, the decoder decodes the video frame using the cached frame reference.

Is this page helpful?

Yes No

2.2.2 RTVideo Basic RTP Payload Format

02/14/2019 • 2 minutes to read

The size of the RTVideo Basic Payload Format header varies. The minimum size is one byte without the codec headers present. If the codec headers are present, the maximum size is 65 bytes.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2
A	B	C	D	E	F	G	H	I (optional)								Codec Headers Bytes (vari						
...																						

A - M (1 bit): Payload format mode. This field MUST be set to zero in the RTVideo Basic RTP Payload Format mode. The field is set to one in other RTP payload formats, as specified in sections 2.2.3, 2.2.4, and 2.2.5.

B - C (1 bit): Cached frame flag. A value of one specifies a cached frame. A value of zero specifies the frame is not a cached frame.

C - SP (1 bit): Super P-frame (SP-frame) flag. A value of one specifies an SP-frame. A value of zero specifies the frame is not an SP-frame.

D - L (1 bit): Last packet flag. Indicates whether this packet is the last packet of the video frame, excluding FEC metadata packets. A value of one specifies the last packet. A value of zero specifies it is not the last packet.

E - O (1 bit): MUST be set to one.

F - I (1 bit): I-frame flag. Indicates whether the frame is an I-frame. A value of one indicates the frame is an I-frame. A value of zero indicates that the frame is not an I-frame, but rather a SP-frame, P-frame, or B-frame.

2.2.3 RTVideo Extended RTP Payload Format

02/14/2019 • 5 minutes to read

The size of the RTVideo Extended RTP Payload Format header varies. The minimum size is 4 bytes without the codec headers present. With codec headers present, the maximum size is 68 bytes.

The frame counters described in the following paragraphs are meaningful only within a video GOP. The counter starts at zero for the first frame in a GOP and increments by one for every succeeding frame. The frame counter is reset to zero at the beginning of the next GOP.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2
A	B	C	D	E	F	G	H	I	J		HiFC		DV		K	FrameCounter						
L (optional)								Codec Headers Bytes (variable)														
...																						

A - M (1 bit): Payload format mode. It MUST be set to one in the RTVideo Extended RTP Payload Format.

B - C (1 bit): Cached frame flag. A value of one specifies a cached frame. A value of zero specifies the frame is not a cached frame. The decoder on the receiver side MUST cache the cached frame because the next SP-frame references it.

C - SP (1 bit): Super P-frame (SP-frame) flag. A value of one specifies an SP-frame. A value of zero specifies the frame is not an SP-frame.

2.2.4 RTVideo Extended 2 RTP Payload Format

02/14/2019 • 5 minutes to read

The size of the RTVideo Extended 2 RTP Payload Format header varies. The minimum size is 8 bytes without codec headers present. With codec headers present, the maximum size is 72 bytes.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2
A	B	C	D	E	F	G	H	I	J		HiFC		DV		K	FrameCounter						
Reserved																						
L (optional)								Codec Headers Bytes (variable)														
...																						

A - M (1 bit): Payload format mode. It MUST be set to one in the RTVideo Extended 2 RTP Payload Format.

B - C (1 bit): Cached frame flag. A value of one specifies a cached frame. A value of zero specifies the frame is not a cached frame. The decoder on the receiver side MUST cache the cached frame because the next SP-frame references it.

C - SP (1 bit): Super P-frame (SP-frame) flag. A value of one specifies an SP-frame. A value of zero specifies the frame is not an SP-frame.

D - L (1 bit): Last packet flag. Indicates whether this packet is the last packet of the video frame, excluding FEC metadata packets. A value of one specifies the last packet. A value of zero specifies that it is not the last packet.

E - O (1 bit): MUST be set to one.

Attachment 13

Etheridge Law Group, PLLC
2600 East Southlake Blvd
Suite 120 / 324
Southlake, TX 96092

November 15, 2019

Counsel for Google (via email)

RE: *Uniloc 2017 LLC et al. v. Google LLC,* Case No. 2:18-cv-00501

Counsel:

We write to address Google's deficient document productions in the above listed case. In particular, Google has not produced the requisite technical documents it was affirmatively obligated to produce under the Court's orders, the local rules, and this District's case law governing the same.

Uniloc continues to evaluate and analyze the deficiencies in Google's productions and may identify further deficiencies in the future. But given the continuing case schedule and upcoming deadlines, Uniloc requires resolution of these immediate issues in a timely fashion to avoid further prejudice to Uniloc.

Obligations Under P.R. 3-4(a)

Pursuant to the Docket Control Order, Google's P.R. 3-4(a) required a production of:

> "(a) Source code, specifications, schematics, flow charts, artwork, formulas, or other documentation sufficient to show the operation of any aspects or elements of an Accused Instrumentality identified by the patent claimant in its P. R. 3-1(c) chart"

The Eastern District of Texas has provided guidance on such P.R. 3-4(a) productions:

> P. R. 3-4(a) requires [a defendant] to produce more than the bare minimum of what *it* believes is sufficient, including but not limited to any and all source code, specifications, schematics, flow charts, artwork, formulas, or other documentation in its possession.

IOLI Trust v. Avigilon Corp. No. 2:10-CV-605-JRG. E.D. TX Nov. 16, 2012)(emphasis in original).

> . . . P. R. 3-4(a) imposes an affirmative obligation on the accused infringer to produce source code and all other relevant materials reasonably needed for the Plaintiff to understand *for itself* how the technology at issue operates and functions

Id. Google's compliance with P.R. 3-4(a) is long overdue.

Obligations Pursuant to the Court's Discovery Order

Also, pursuant to the discovery order in this case, Google was also required to produce relevant documents without waiting for a discovery request. Judge Mazzant recently reiterated this District's long-standing practice concerning such a procedure. *See Natural Polymer International Corp. v. The Hartz Mountain Corp., Case No.* 4:18-cv-667, Dkt. No. 27 (E.D. Tex.)(June 20, 2019)("Therefore, the Court reminds the parties of their duty to produce all relevant information as it is discovered, without waiting for formal discovery requests.").

Notwithstanding this affirmative obligation without a request, Uniloc sent Google a letter on October 15, 2019 specifically identifying, among other things, a variety of technical documents it expected to be produced.

Google's lack of technical production is illustrated by its interrogatory answers concerning non-infringement positions. Uniloc specifically sought documents supporting any such position. However, Google answered cited no documents, but rather indicated that it will produce documents and later identify support. If Google had already complied with its production requirements, Google could have simply identified its production.

Reasonably Similar Systems

In correspondence between the parties, Google also admits that it has not produced documents concerning reasonably similar systems. Google alleges that it need not produce these reasonably similar systems because they are not charted. However, Google's position has been explicitly rejected by the Eastern District of Texas. The Eastern District of Texas places an affirmative obligation on an accused infringer to produce documents concerning "reasonably similar" to accused instrumentalities – regardless of whether such an item is accused. *See Epicrealm Licensing, LLC v. Autoflex Leasing, Inc.*, Nos. 2:05-CV-163-DF-CMC, 2:05-CV-356-DF-CMC, 2007 WL 2480969, at *3 (E.D. Tex. Aug. 27, 2007) ("discovery in a patent infringement suit 'includes discovery relating to the technical operation of the accused products, as well as the identity of and technical operation of any products reasonably similar to any accused product.'"); *DDR Holdings, LLC v. Hotels.com, L.P.*, No. 2:06-CV-42-JRG, 2012 WL 2935172, at *10-11 (E.D. Tex. July 18, 2012) ("It is well settled in the Eastern District that 'there is no brightline rule that discovery is permanently limited to the products specifically accused in a party's [infringement contentions].' Such a limitation would be 'inconsistent with the broad discovery regime created by the Federal Rules and the notion that a party may be able to amend its [infringement contentions.]' Therefore, discovery may be properly extended to products 'reasonably similar' to those accused in [infringement contentions].") (internal citations omitted).

We encourage you to carefully read the *EpicRealm Licensing* case where the Defendant made the same arguments that Google makes here. Uniloc apprised Google of the above-cases in its infringement contentions served months ago.

Given the core importance of these technical documents, including use within the claim construction process that is already underway, we ask for immediate resolution of these issues and complete production of these documents before November 22. In you intend to continue withholding documents, Court intervention will be necessary, and we request a Meet and Confer next week. Please provide a couple windows of time when Lead and Local counsel are available.

Best regards,

<u>/s/ Ryan S. Loveless</u>

Ryan S. Loveless

Attachment 14

Etheridge Law Group, PLLC
2600 East Southlake Blvd
Suite 120 / 324
Southlake, TX 96092

November 15, 2019

Counsel for Google (via email)

RE: *Uniloc 2017 LLC et al. v. Google LLC,* Case No. 2:18-cv-00551

Counsel:

We write to address Google's deficient document productions in the above listed case. In particular, Google has not produced the requisite technical documents it was affirmatively obligated to produce under the Court's orders, the local rules, and this District's case law governing the same.

Uniloc continues to evaluate and analyze the deficiencies in Google's productions and may identify further deficiencies in the future. But given the continuing case schedule and upcoming deadlines, Uniloc requires resolution of these immediate issues in a timely fashion to avoid further prejudice to Uniloc.

Obligations Under P.R. 3-4(a)

Pursuant to the Docket Control Order, Google's P.R. 3-4(a) required a production of:

> "(a) Source code, specifications, schematics, flow charts, artwork, formulas, or other documentation sufficient to show the operation of any aspects or elements of an Accused Instrumentality identified by the patent claimant in its P. R. 3-1(c) chart"

The Eastern District of Texas has provided guidance on such P.R. 3-4(a) productions:

> P. R. 3-4(a) requires [a defendant] to produce more than the bare minimum of what *it* believes is sufficient, including but not limited to any and all source code, specifications, schematics, flow charts, artwork, formulas, or other documentation in its possession.

IOLI Trust v. Avigilon Corp. No. 2:10-CV-605-JRG. E.D. TX Nov. 16, 2012)(emphasis in original).

> . . . P. R. 3-4(a) imposes an affirmative obligation on the accused infringer to produce source code and all other relevant materials reasonably needed for the Plaintiff to understand *for itself* how the technology at issue operates and functions

Id. Google's compliance with P.R. 3-4(a) is long overdue.

Obligations Pursuant to the Court's Discovery Order

Also, pursuant to the discovery order in this case, Google was also required to produce relevant documents without waiting for a discovery request. Judge Mazzant recently reiterated this District's long-standing practice concerning such a procedure. *See Natural Polymer International Corp. v. The Hartz Mountain Corp., Case No.* 4:18-cv-667, Dkt. No. 27 (E.D. Tex.)(June 20, 2019)("Therefore, the Court reminds the parties of their duty to produce all relevant information as it is discovered, without waiting for formal discovery requests.").

Notwithstanding this affirmative obligation without a request, Uniloc sent Google a letter on October 15, 2019 specifically identifying, among other things, a variety of technical documents it expected to be produced.

Google's lack of technical production is illustrated by its interrogatory answers concerning non-infringement positions. Uniloc specifically sought documents supporting any such position. However, Google answered cited no documents, but rather indicated that it will produce documents and later identify support. If Google had already complied with its production requirements, Google could have simply identified its production.

Reasonably Similar Systems

In correspondence between the parties, Google also admits that it has not produced documents concerning reasonably similar systems. Google alleges that it need not produce these reasonably similar systems because they are not charted. However, Google's position has been explicitly rejected by the Eastern District of Texas. The Eastern District of Texas places an affirmative obligation on an accused infringer to produce documents concerning "reasonably similar" to accused instrumentalities – regardless of whether such an item is accused. *See Epicrealm Licensing, LLC v. Autoflex Leasing, Inc.*, Nos. 2:05-CV-163-DF-CMC, 2:05-CV-356-DF-CMC, 2007 WL 2480969, at *3 (E.D. Tex. Aug. 27, 2007) ("discovery in a patent infringement suit 'includes discovery relating to the technical operation of the accused products, as well as the identity of and technical operation of any products reasonably similar to any accused product.'"); *DDR Holdings, LLC v. Hotels.com, L.P.*, No. 2:06-CV-42-JRG, 2012 WL 2935172, at *10-11 (E.D. Tex. July 18, 2012) ("It is well settled in the Eastern District that 'there is no brightline rule that discovery is permanently limited to the products specifically accused in a party's [infringement contentions].' Such a limitation would be 'inconsistent with the broad discovery regime created by the Federal Rules and the notion that a party may be able to amend its [infringement contentions.]' Therefore, discovery may be properly extended to products 'reasonably similar' to those accused in [infringement contentions].") (internal citations omitted).

We encourage you to carefully read the *EpicRealm Licensing* case where the Defendant made the same arguments that Google makes here. Uniloc apprised Google of the above-cases in its infringement contentions served months ago.

Given the core importance of these technical documents, including use within the claim construction process that is already underway, we ask for immediate resolution of these issues and complete production of these documents before November 22. In you intend to continue withholding documents, Court intervention will be necessary, and we request a Meet and Confer next week. Please provide a couple windows of time when Lead and Local counsel are available.

Best regards,

/s/ Ryan S. Loveless

Ryan S. Loveless

Attachment 15

Etheridge Law Group, PLLC
2600 East Southlake Blvd
Suite 120 / 324
Southlake, TX 96092

November 15, 2019

Counsel for Google (via email)

RE: *Uniloc 2017 LLC et al. v. Google LLC,* Case No. 2:18-cv-00497

Counsel:

We write to address Google's deficient document productions in the above listed case. In particular, Google has not produced the requisite technical documents it was affirmatively obligated to produce under the Court's orders, the local rules, and this District's case law governing the same.

Uniloc continues to evaluate and analyze the deficiencies in Google's productions and may identify further deficiencies in the future. But given the continuing case schedule and upcoming deadlines, Uniloc requires resolution of these immediate issues in a timely fashion to avoid further prejudice to Uniloc.

Obligations Under P.R. 3-4(a)

Pursuant to the Docket Control Order, Google's P.R. 3-4(a) required a production of:

> "(a) Source code, specifications, schematics, flow charts, artwork, formulas, or other documentation sufficient to show the operation of any aspects or elements of an Accused Instrumentality identified by the patent claimant in its P. R. 3-1(c) chart"

The Eastern District of Texas has provided guidance on such P.R. 3-4(a) productions:

> P. R. 3-4(a) requires [a defendant] to produce more than the bare minimum of what *it* believes is sufficient, including but not limited to any and all source code, specifications, schematics, flow charts, artwork, formulas, or other documentation in its possession.

IOLI Trust v. Avigilon Corp. No. 2:10-CV-605-JRG. E.D. TX Nov. 16, 2012)(emphasis in original).

> . . . P. R. 3-4(a) imposes an affirmative obligation on the accused infringer to produce source code and all other relevant materials reasonably needed for the Plaintiff to understand *for itself* how the technology at issue operates and functions

Id. Google's compliance with P.R. 3-4(a) is long overdue.

Obligations Pursuant to the Court's Discovery Order

Also, pursuant to the discovery order in this case, Google was also required to produce relevant documents without waiting for a discovery request. Judge Mazzant recently reiterated this District's long-standing practice concerning such a procedure. *See Natural Polymer International Corp. v. The Hartz Mountain Corp., Case No.* 4:18-cv-667, Dkt. No. 27 (E.D. Tex.)(June 20, 2019)("Therefore, the Court reminds the parties of their duty to produce all relevant information as it is discovered, without waiting for formal discovery requests.").

Notwithstanding this affirmative obligation without a request, Uniloc sent Google a letter on October 15, 2019 specifically identifying, among other things, a variety of technical documents it expected to be produced.

Google's lack of technical production is illustrated by its interrogatory answers concerning non-infringement positions. Uniloc specifically sought documents supporting any such position. However, Google answered cited no documents, but rather indicated that it will produce documents and later identify support. If Google had already complied with its production requirements, Google could have simply identified its production.

Reasonably Similar Systems

In correspondence between the parties, Google also admits that it has not produced documents concerning reasonably similar systems. Google alleges that it need not produce these reasonably similar systems because they are not charted. However, Google's position has been explicitly rejected by the Eastern District of Texas. The Eastern District of Texas places an affirmative obligation on an accused infringer to produce documents concerning "reasonably similar" to accused instrumentalities – regardless of whether such an item is accused. *See Epicrealm Licensing, LLC v. Autoflex Leasing, Inc.*, Nos. 2:05-CV-163-DF-CMC, 2:05-CV-356-DF-CMC, 2007 WL 2480969, at *3 (E.D. Tex. Aug. 27, 2007) ("discovery in a patent infringement suit 'includes discovery relating to the technical operation of the accused products, as well as the identity of and technical operation of any products reasonably similar to any accused product.'"); *DDR Holdings, LLC v. Hotels.com, L.P.*, No. 2:06-CV-42-JRG, 2012 WL 2935172, at *10-11 (E.D. Tex. July 18, 2012) ("It is well settled in the Eastern District that 'there is no brightline rule that discovery is permanently limited to the products specifically accused in a party's [infringement contentions].' Such a limitation would be 'inconsistent with the broad discovery regime created by the Federal Rules and the notion that a party may be able to amend its [infringement contentions.]' Therefore, discovery may be properly extended to products 'reasonably similar' to those accused in [infringement contentions].") (internal citations omitted).

We encourage you to carefully read the *EpicRealm Licensing* case where the Defendant made the same arguments that Google makes here. Uniloc apprised Google of the above-cases in its infringement contentions served months ago.

Given the core importance of these technical documents, including use within the claim construction process that is already underway, we ask for immediate resolution of these issues and complete production of these documents before November 22. In you intend to continue withholding documents, Court intervention will be necessary, and we request a Meet and Confer next week. Please provide a couple windows of time when Lead and Local counsel are available.

Best regards,

/s/ Ryan S. Loveless

Ryan S. Loveless

Attachment 16

Attachment 17